

DeepView: A Channel for Distributed Microscopy and Informatics

B. Parvin, J. Taylor, G. Cong

Information and Computing Sciences Division
Lawrence Berkeley National Laboratory
Berkeley, CA 94720
<http://vision.lbl.gov>

M. A. OKeefe

Material Sciences Division
Lawrence Berkeley National Laboratory
Berkeley, CA 94720

M.H. Barcellos-Hoff

Life Sciences Division
Lawrence Berkeley National Laboratory
Berkeley, CA 94720

September 8, 1999

Abstract

This paper outlines the requirements, architecture, and design of a “Microscopy Channel” over the wide area network. A microscopy channel advertises a listing of available online microscopes, where users can seamlessly participate in an experiment, acquire expert opinions, collect and process data, and store this information in their electronic notebook. The proposed channel is a collaborative problem solving environment (CPSE) that allows for both synchronous and asynchronous collaboration.

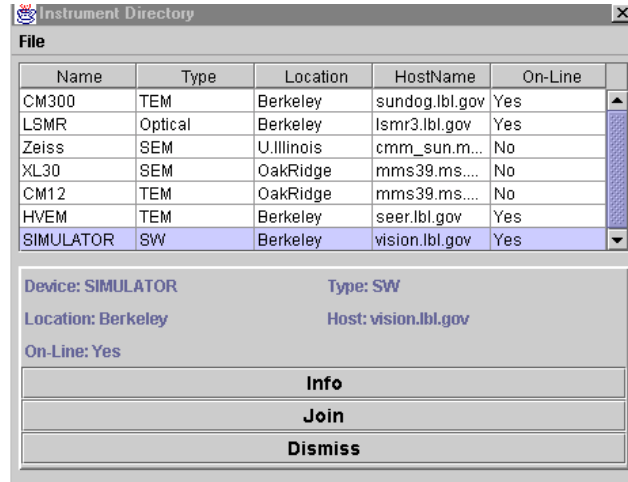
Our testbed includes several unique electron and optical microscopes with applications ranging from material science to cell biology. We have studied current commercial CORBA services and concluded that three basic services are needed to meet the extensibility and functionality constraints. These include: Instrument Services (IS), Exchange Services (ES), and Computational Services (CS). These services sit on top of CORBA and its enabling services (naming, trading, security, and notification). IS provide a layer of abstraction for controlling any type of microscope. ES provide a common set of utilities for information management and transaction. CS provide the analytical capabilities needed for online microscopy and PSE.

*This work is supported by the Director, Office of Science, Office of Advanced Scientific Computing Research, Mathematical, Information, and Computational Sciences Division, Office of Basic Energy Service; and Assistant Secretary of Environmental Research, Office of Biological and Environmental Research of the U. S. Department of Energy under Contract No. DE-AC03-76SF00098 with the University of California. The LBNL publication number is 43557.

1 Introduction

The current trend in telepresence research is to bring experts and facilities together from geographically dispersed locations [6, 13, 17, 21]. The natural evolution of this research is to construct a scalable system for collaboration and to leverage either legacy or new computational toolkits to support novel scientific applications based on capabilities in simulation, inverse problem solving, visualization, real-time control, and steering. The intent is to build a domain-specific problem-solving environment. The Department of Energy is developing such an environment as a part of its DOE2000 National Collaboratory Program.

The themes for the proposed architecture are functionality, scalability, and performance. We are also interested in interactivity, which is achieved through the best effort with most commercial ORBs. Functionality refers to what and how an instrument does something and how well system resources can be managed and accessed. Scalability refers to the number of instruments, vendor-specific desktop workstations, analysis programs, and collaborators that can simultaneously attach themselves to the system. Performance refers to how well system resources are being utilized. Our testbed includes several electron and optical microscopes that are located at Berkeley Lab(LBNL), Oak Ridge National Laboratory(ORNL), and the University of Illinois with applications ranging from material science to cell biology. The interface to the microscopy channel along with a listing of various instruments are shown in Figures 1 and 2. The channel is tightly coupled with the OMG-defined[5] Naming and Trading Services for binding and resource discovery. From the user's perspective, we establish a set of desirable requirements in terms of functionality, scalability, interactivity, safety, and security. From the designer's perspective, we abstract these requirements into three categories of services: Instrument Services (IS), Exchange Services (ES), and Computational Services (CS). These services sit on top of CORBA and its enabling services. IS provides a layer of abstraction for controlling any type of microscope or simulation software. Simulation aims at generating a representation based on physical properties of a system and its relationship with respect to an observation mode. ES provides a common set of utilities for information management and transaction. CS provides the analytical capabilities needed for online microscopy and problem solving. The design also maximizes the use of existing off-the-shelf software components.



The screenshot shows a window titled "Instrument Directory" with a "File" menu. It contains a table with columns: Name, Type, Location, HostName, and On-Line. The table lists several instruments, with "SIMULATOR" selected. Below the table, the details for the selected instrument are shown: Device: SIMULATOR, Type: SW, Location: Berkeley, Host: vision.lbl.gov, and On-Line: Yes. At the bottom, there are three buttons: Info, Join, and Dismiss.

Name	Type	Location	HostName	On-Line
CM300	TEM	Berkeley	sundog.lbl.gov	Yes
LSMR	Optical	Berkeley	lsmr3.lbl.gov	Yes
Zeiss	SEM	U.Illinois	cmr_sun.m...	No
XL30	SEM	OakRidge	mms39.ms...	No
CM12	TEM	OakRidge	mms39.ms...	No
HVEM	TEM	Berkeley	seer.lbl.gov	Yes
SIMULATOR	SW	Berkeley	vision.lbl.gov	Yes

Device: SIMULATOR Type: SW
Location: Berkeley Host: vision.lbl.gov
On-Line: Yes

Info
Join
Dismiss

Figure 1: User's view of the microscopy channel.

CS offers an extensible array of tools for visualization, model recovery, and comparative analysis of observed and simulated data. Model recovery is an inverse problem-solving process that attempts to (a) link a specimen’s behavior to external stimulation through feature extraction, archival, and data mining or (b) construct a 3D geometric model of an object through user interaction. In general, model recovery is a computation-intensive algorithmic process requiring the extensive support of high-performance computing and low-latency network infrastructure.

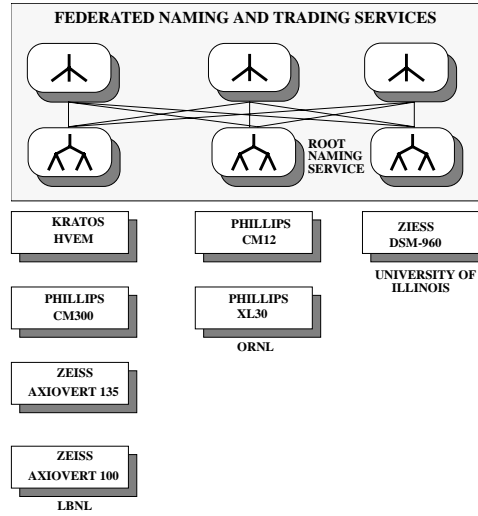


Figure 2: Listing of instruments at three different sites. Instruments, their classification, and their required resources are registered with an OMG-defined federated Naming and Trading Service.

The next section of the paper summarizes recent related work in collaborative computing. Section 3 describes software architecture, ongoing scientific experiments and their corresponding computational needs. Section 4 concludes with a summary of the versatility of this architecture.

2 Related Work

Previous systems fall under two categories: telepresence [6, 7, 13, 16] and collaborative frameworks. Telepresence research has focused on remote functionality of the instrument and the necessary automation for large-scale-data collection and analysis. In general, these systems ignore many of the scalability issues that we have been advocating [14, 15]. With respect to the collaborative framework, a taxonomy of existing systems is given below.

- UC Berkeley’s MASH project [9] uses MBone tools in a heterogeneous environment to develop scalable multi-media architecture for collaborative applications in fully distributed systems.
- NCSA’s Habanero project provides smooth management and simultaneous distribution of shared information to all clients in a component-based, centralized system written primarily in Java.
- Rutgers University’s DISCIPLE uses a CORBA framework for distributed access in a service-based, centralized system for enforcing shared virtual space.

- Sun Microsystem's Java Shared Development tool kit enables collaborative-aware Java code to send data to participants within a communication session. It supports three types of transport protocols: TCP/IP socket, light-weight reliable multicast, and remote invocation method. In this framework, all objects are manageable and collaboration occurs within a session that includes channel, token, blobs, and listener.
- The University of Michigan's Upper Atmosphere Research Collaboratory (UARC) [22] is a web-based distributed system (written mostly in Java) that collects data from over 40 observational platforms for space physics research for both synchronous and asynchronous collaboration. In this system, data suppliers publish their data on a data-dissemination server. Clients then subscribe to receive the desired information.

Our approach is to construct a service-based framework that is distributed, extensible, and maximizes the use of common off-the-shelf (COTS) software to exploit the economy of scales. This is based on an OMG-defined CORBA framework [with an Internet Inter-ORB Protocol (IIOP)] that is supported by Netscape Communicator. CORBA provides virtual distributed containers for objects. These objects can then be implemented in any language, e.g., Java, C++. Furthermore, OMG has defined a number of enabling technologies for decoupled communication, object localization and resource discovery, and security. The main advantages of using CORBA are that (a) it is not restricted to the Java language, (b) it is available on multiplatforms, (c) it supports a rich class of enabling services, and (d) it supports real-time applications [19] under a newly adopted standard.

3 Software Architecture

Our system uses an extensible object-oriented framework (class libraries, APIs, and shared services) so that applications can be rapidly assembled, maintained, and reused. These objects may reside on any host and can be listed, queried, and activated in the system. The architecture illustrated in Figure 3 bridges the gaps between different services that may reside at any node in a distributed system.

Our system consists of three service categories that interact with the ORB: Instrument Services (IS), Exchange Services (ES), and Computational Services (CS). These core services are specified in IDL. A brief review of Enabling Services is provided in this section.

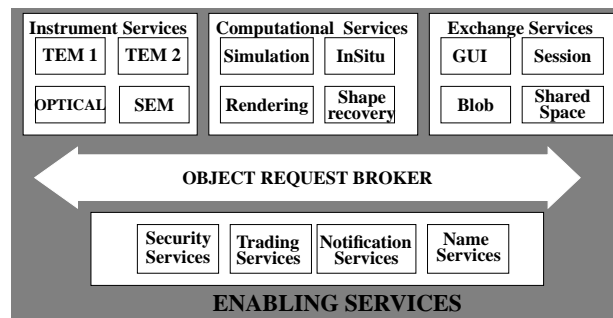


Figure 3: Interaction Between OMG-defined Enabling Services and proposed Services.

3.1 Enabling Services

Our system uses Naming, Trading, Security, and Notification Services. The Naming Service binds a name to an object and allows that object to be found subsequently. It behaves much like the White Pages. The names are resolved within a naming context that is organized as a graph. The naming context is an object that stores name binding for objects, and it is essentially a table. The Trading Service provides facilities for dynamic object discovery. The trader stores a description of the service along with object reference, and behaves much like the Yellow Pages. It provides an advertisement service, policies, and a matching engine through an OMG-defined constraint language. The constraint is a boolean expression that is somewhat similar to an SQL interface. The constraint language provides boolean, arithmetic, and comparison operations to locate a particular object or resource based on its properties. The requirement for resource management and brokering has been well documented [1, 18]. For example, Globus uses a metacomputing directory service based on LDAP, and the San Diego Super Computing Center has developed its metadata on top of a relational database, providing additional flexibility and better search capability. The CORBA Naming and Trading Services are an alternative approach that is reliable (for writing), lightweight, and well supported by the commercial sector. While the Naming Service is hierarchical (much like UNIX file system), the Trading Service is flat. The Security Service is based on the secure socket layer (SSL), which provides authentication, privacy, and integrity for TCP-based connections. SSL uses RSA public key cryptography for authentication, where each application has an associated public key and an associated private key. In this context, data encrypted with the public key can only be decrypted with the private key, and data encrypted with the private key can only be decrypted with the public key. The Notification Service is a newly defined OMG standard that replaces the Event Services. OMG has defined two basic models of communication:

- The first model is based on the standard CORBA invocation model of two-way, one-way, and deferred synchronous interaction. Although this model simplifies distributed processing, it lacks asynchronous message delivery and does not support group communication that can lead to excessive polling.
- The second model uses COS Event Services that provide decoupled communication between suppliers and consumers. The key concept in this service is the Event Channel, which can assume a variety of design patterns depending on the model of collaboration among different components [20]. The roles that the Event Channel can play include (a) a notifier for the push/push model, (b) a procurer for the pull/pull model, (c) a queue for the hybrid push/pull model, and (d) an intelligent agent for the hybrid pull/push model. Presently, only the push/push model with typed and untyped events is supported. This service allows clients to register with events of interest and filter incoming events. A unique feature of the Notification Service is that it supports quality of service (QoS) and various policies to enforce it. The underlying transport protocol can be either TCP or reliable multicast. With reliable multicast, the event channel is replaced with smart proxies. Presently, only Untyped data are supported with COS Event Services API through by OrbixTalk from Iona. OrbixTalk provides assembly, sequencing, and ordering of IP multicast packets for enhanced network utilization. The COS Event API allows any IIOP client to make use of multicast functionality.

In our system, all synchronous communications (for collaboration) are performed through the event channel, and all asynchronous operations are conducted with the CORBA two-way and one-way communication. The Naming and Trading Service is used for object localization and its required resources. The Naming Service is organized as a hierarchical tree structure for modular organization of objects. These services are federated with each physical site maintaining its own catalog of information. However, this view is hidden from clients. The key advantages of a federated organization are (a) improved reliability (when a single server becomes inaccessible), (b) improved

performance (where different servers can work in parallel), (c) improved scalability (where persistent information is distributed on multiple hosts), and (d) improved administration boundaries. Naming and Trading Services are a powerful mechanism for resource discovery, brokering, and subsequent load balancing.

The SSL handshake is initiated by a client sending a message to the server. The server responds by sending its X.509 certificate. The client extracts the public key from the certificate and encrypts a session key. The server uses its private key to decrypt the session key and application data. Additionally, the server requests the client certificate to resume a previously established handshake.

3.2 Instrument Services

Instrument Services provide a scalable means of collaborative instrument control and interaction through three objects: instrument, instrument factory, and an abstract action class. See Figure 4. An instrument consists of a set of devices (e.g., controller, detectors) that are advertised through the Naming Service. Each device has a list of properties. For example, the controller may include focus, shift, and tilt properties. These properties and their corresponding attributes can be queried and manipulated through instances of instrument and action objects. An action consists of three simple interfaces: get, set, and cando. The action is polymorphic for actions taking place between the local area network (LANAction) and the wide area network (WANAction). A LANAction is implemented as an efficient control mechanism to avoid SSL handshake for near real-time control.

All actions occur within a managed session. When clients instantiate an action, they pass an object that uniquely identify themselves. By associating a particular user with each action, the server may queue and/or prioritize the processing of its services in the collaborative environment. The design of IS is partially influenced by the Object Property Service (OPS) as defined by OMG. OPS provides a mechanism to associate objects with typed Name-Value pairs. These objects can then be manipulated through a set and get methods. In an instrument, the value of a control parameter can change either by natural drifts in the system or when it is set to a new value by a principal client. The design of the server is multi-threaded. One thread autonomously scans the property values (of each device) every n seconds. The second thread simply changes the value of a property through a set operation. In both cases, any changes in the state of the system are recorded and then broadcast to all clients. This design is scalable since properties and their corresponding attributes are stored in a persistent configuration file. In other words, addition of new instruments will not require any changes to the CORBA IDL representation, and thus no changes will be needed to the clients (with the exception of instrument-specific changes to the GUI). Each new instrument needs to define a “plug-in” for the proposed architecture. Our experience indicates that an open instrument (with documented DLL or COM interface) can easily be interfaced to this system. Another important issue is incorporation of the vendor’s software into the baseline system. This is significant in terms of functionality, because microscopists use these tools as a part of their routine experiment. The CORBA-COM bridge allows those functionalities to be present in our system. Functionality of vendor’s software is often achieved through a scripting language.

Another utility of IS is its use as a front end to a simulation engine or modeling system. In this context, a physical instrument and a simulation engine behave similarly. They both have control parameters, and they both generate blobs of data.

3.3 Exchange Services

Exchange Services provide a set of objects for information exchange between collaborators, instruments, and application programs. This set consists:

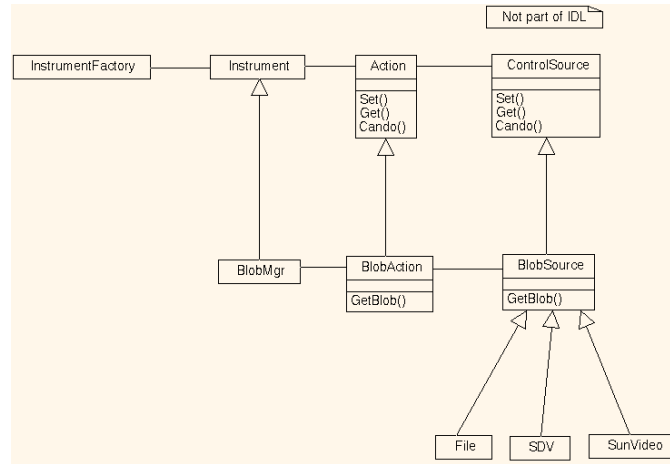


Figure 4: Key objects for Blob Manager and Instrument Manager.

- The session manager (SessionMgr) object provides a listing of active users and a policy for sharing an instrument among multiple collaborators. This policy empowers the current “principal” to pass the control to another user. The instrument has a local operator who can override current the principal by assigning the instrument to a third party. Each time a new user joins the system, his or her presence is broadcast to all the other clients. Likewise, when the user leaves, he or she is removed from the list of active clients.
- The blob manager (BlobMgr) object provides an efficient means of transferring bulk data between various objects. It uses the same IDL that is used by IS, but it is extended to handle blob data. The recent implementation of ORBs from Iona and TAO has zero memory copy and has similar performance to the UNIX Socket over the high-speed network. This has significantly simplified our current implementation for bulk data transfer, which was limited to a fixed-size array in the past. Over the wide area network, most of the bulk data transfer is compressed (with either JPEG or Wavelet) and implemented as a sequence data type. The server side of the blob manager is designed in such a way that various sources of blob objects (detectors) are hidden from the IDL. As a result, each time a new detector (blob generator) is added to the system, no changes to the IDL are made. A client can query various blob sources in the system and select one to receive the data. The blob object is managed by the session manager.
- The shared space manager provides the necessary services for clients to exchange chat messages, graphics overlays, and images among multiple collaborators. Message sharing can be private or public. Public messages are broadcast to all collaborators, while private messages are sent to a subset of collaborators. This component is tightly coupled to the session manager for private messages.

In our system, collaborators communicate with servers through OrbixWeb (a Java version of Iona’s ORB), where remote GUI objects are implemented as Java beans. In theory, Java provides scalability on different types of desktops. However, some modification is needed for porting the GUI across multiple platforms. The GUI component aims to provide needed functionality for a particular type of experiment. A fairly detailed abstraction exists for in-situ and high-resolution microscopy. It also provides a log-book where the state of the instrument can be traversed to a

previously known state. The GUI manager has two components: a generic interface for common instrument control, and a set of specialized components that are instrument specific.

The Java client is based on the Model View Controller (MVC) pattern. The implementation uses the Java Event Model. The goal is to allow some clients to receive synchronous notification, while others rely on an optimistic policy. A synchronous approach can be used to view the dynamic state of an instrument, while an optimistic policy is used strictly for listening. The Model provides a proxy for a remote data source, and provides three kinds of interfaces: Command, Data, and Notification. The Command interface is for requesting changes to the Model's data. The Data interface is for requesting the Model's data. The Notification interface is for notifying listeners about changes in the Model's data as a result of external events. The key Models (event sources) in the DeepView client are BlobSource, ControlSource, MessageSource, and SessionSource. Each of these sources can generate user-defined events. The View components receive input from the user and forward it to an appropriate controller. It displays data to the user by requesting it from a Model. Additionally, the View receives notification from the Model. The basic Views in the system are BlobCanvas, Device and PropertyTable, and SessionPanel. The Controller defines the relationship between input from the View and the action made against the Model. It follows an Observer pattern. The Controllers use the three interfaces provided by the Model to maintain a consistent View. The Controllers are BlobPlayer, Instrument Manager, and Session Manager.

Figure 5 shows the generic GUI that is used to view of the blob manager, session manager, shared space manager, and the instrument manager. Figure 6 shows the interaction between clients and various services through the Event Channel. At the instrument site, there are three channels for Instrument State, Blob Manager, and Session Manager. The Blob Manager samples the output of the detector periodically and pushes a compressed image to the event channel. This is the most active channel. The other two channels simply notify the client application of any changes. This design is service based, decoupled, and distributed. The behavior of the event channel indicates that it broadcasts data at the rate of the client with the least amount of network bandwidth. Thus, to avoid penalizing clients with high network bandwidth, the system maintains a pool of event channels for the Blob Manager. Each channel has a different updating frequency, and its characteristic is registered with the Trading Service. The clients then measure their bandwidth and connect themselves to an appropriate channel with matching impedance.

3.4 System Views

DeepView maintains three views of the system: the Naming View, Content View, and Meta View. Naming View leverages the vendor-supported GUI for advertising object references and their corresponding hierarchy, which are available in a distributed network. Content View shows the current state of the instrument in terms of devices and their associated properties in the instrument control panel, as shown in Figure 5. Meta View uses XML to represent the relationship between devices, properties, and their attributes. This view is used to annotate actions performed on the instrument and for logging information into the archival system.

3.5 Computational Services

A large number of analytical and simulation software products exist for microscopy and microanalysis. We plan to provide the means to integrate vendor's software to our baseline platform. However, several unique computational components have been integrated to accentuate scientific problem solving within the collaborative framework. These include in-situ electron microscopy [13], recovery of 3D shape through holographic microscopy [2, 3], image simulation for high-resolution transmission electron microscopy (HRTEM) [10], and the effect of low-dose radiation on

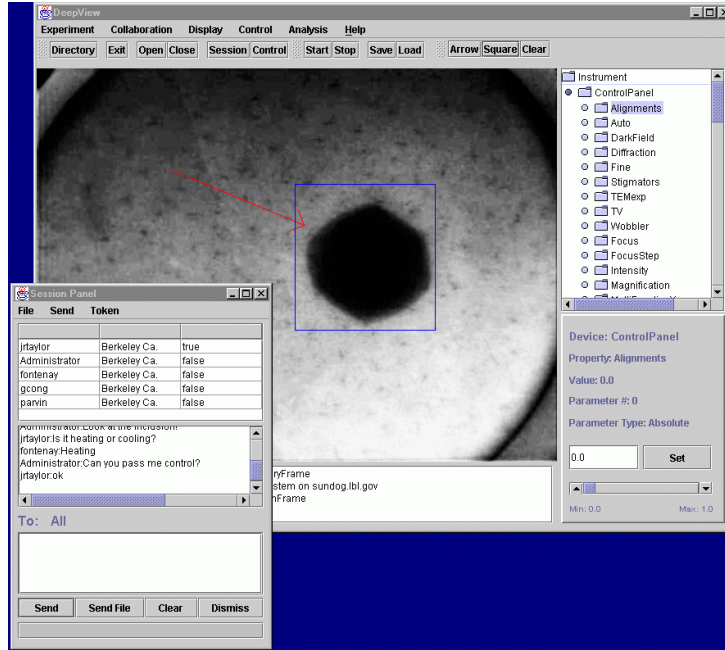


Figure 5: Collaborative view of the MMC shows the shared view space, which includes image, unique graphic color assigned to each collaborator, session manager, and a listing of devices and properties at the instrument site. Changes in the properties are recorded and broadcast to all clients.

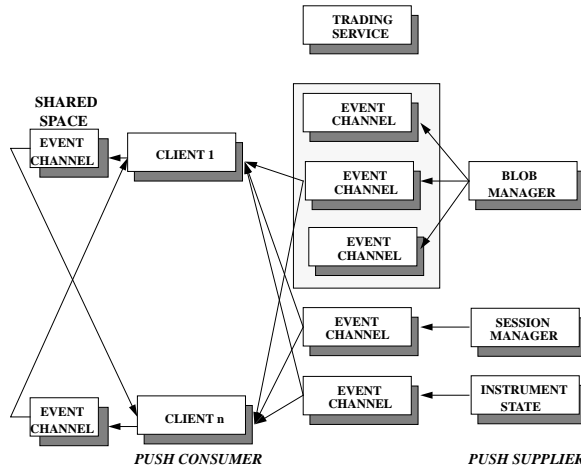


Figure 6: Interaction between producers and consumers through Event Channels. Each consumer measures its available bandwidth and connects itself to a blob manager event channel with matching impedance. The data rate for session manager and instrument manager is low and no impedance matching is needed. The underlying transport for the Event Channel can be either TCP or reliable multicast.

mammary cells. Although objectives are different, many of the computational components (classes) are being shared among above applications. A brief review follows.

In-situ electron microscopy focuses on the behavior of an inclusion (time-dependent morphological changes) as a function of external stimulation, e.g., changes in temperature and pressure. At high magnification, any kind of stress on a specimen manifests spatial drifts in multiple dimensions. The absence of QoS makes this type of experiment nearly impossible over the wide area network. As a result, computational components are brought close to the experimental setup to analyze the videostream, perform online morphological analysis, and compensate for various anomalies so that images at a remote station remain stationary [11, 13]. In this context, the microscope stage is steered to compensate for thermal drifts. The tracking of inclusion is achieved at 8Hz over the local area network using a symmetric multiprocessing system.

The second type of computational service recovers the 3D shape of an object through holographic electron microscopy, in which a new protocol has been developed to recover the 3D shape of an inclusion from multiple views [2, 3]. Conventional electron microscopy presents projected images with little or no depth information. In contrast, electron holography with coherent illumination provides both magnitude and phase information that can be used to infer object thickness in terms of equal thickness contours (ETCs) from each view of the sample. The holographic images contain interference fringes with spacings (in the best case, down to less than an angstrom) in which interference is between the transmitted and diffracted beams. The results of shape recovery for three views of an object and its reconstruction are shown in shown in Figure 7.

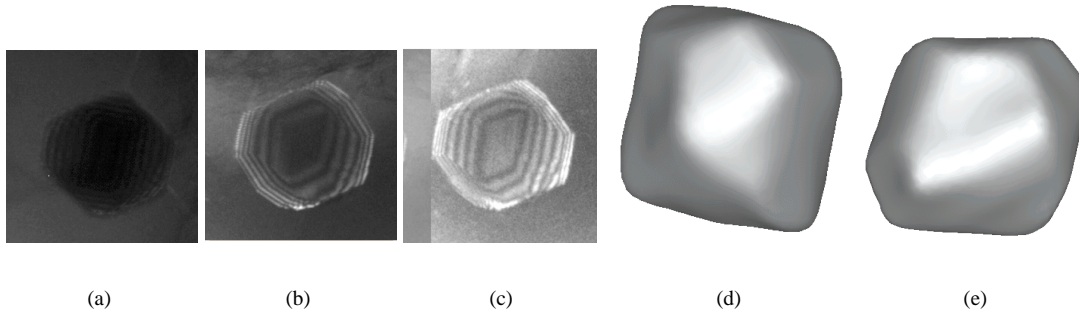


Figure 7: Three views of a cubeoctahedral object with a diameter of 100 nanometers and the corresponding three-dimensional reconstruction.

3.6 Image Simulation

We have incorporated image simulation capability for high-resolution transmission electron microscopy (HRTEM) into our baseline system. This feature provides dynamic comparative analysis of observed and simulated data. The novelty of our design is in how the required parameters for image simulation are represented. This is accomplished by leveraging the same interface that is used by Instrument Services. In this context, image simulation models the electron microscope as well as the specimen. The specimen has three devices: *Atom Configuration*, *Unit Cell*, and *Zone Axis*. The electron microscope has another three devices: *Lens*, *Coherence*, and *Display*. Each of these devices is expressed with its own property list. These devices and properties are then advertised for query and manipulation through

standard get, set, and can-do operations. The image simulation interface has methods for geometric transformation, three-dimensional display of atomic structure, display of microscope lens condition, and image generation at different parameter values. The end result is either a single image that can be rapidly updated or a collage of simulated images computed under different parameter settings (e.g., defocus and thickness). An example of the single image display, superimposed on an experimental result, is shown in Figure 8. Here the user updates the imaging properties to match the current microscope values and watches as the simulation changes. Eventually, the simulation will be configured to read the current microscope values dynamically and to track any changes as they occur. For easy comparison with the experimental image, the user is able to rotate and stretch the simulated image to match the geometry of the experimental one. A typical collage result is shown in Figure 9 as a map of image changes with lens defocus and specimen thickness. Such collages are presented to the user for comparative analysis with observational data. The simulation result along with its annotation text, e.g., devices and properties, are modeled using XML and stored in the archive.

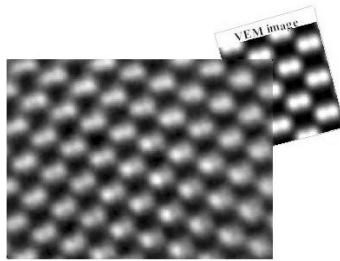


Figure 8: Matching of virtual electron microscope image with observed data through scaling, rotation, and translation.

3.7 Low-Dose Radiation Study

DeepView is being used to study the response of mammary tissue to low-dose ionizing radiation exposure. Here, we are interested in structural and functional mapping of the microanatomy from microenvironment to cellular scale. A typical experimental protocol requires radiated samples to be imaged with different sets of excitation filters for subsequent image analysis, and archival and statistical studies. We have developed a set of computational tools to analyze data and to produce an *intelligent summary* of pertinent features from image space. An example of the localization of proteins and DNA in mammary gland tissue sections is shown in Figure 10. One feature of the mammary microenvironment of particular interest is the basement membrane, a tubular structure that underlies and corresponds to the basal aspect of the mammary cells. The image analysis tool leverages a technique that was developed for the tracking of tubular molecules [12]. The result is shown in Figure 11. The yellow line and red markers correspond to the contour and points of maximum curvature at a particular scale. The blue line corresponds to the medial axis, providing a compact representation of functional activity (radiation-induced protein expression) along the mammary wall.

At the cellular scale, we have developed a protocol to extract the nuclei around the mammary duct. The main issue is how to delineate individual nucleus that are clumped together. Our approach [4] is to extract partial cues corresponding to step and roof edges of the image and then group them through geometric reasoning. Step edges form the boundary between the nuclei, and background and roof edges provide the partial boundary between adjacent nuclei. The intent is to construct a partitioning that is globally consistent. A list of step edges is constructed by thresholding raw images,

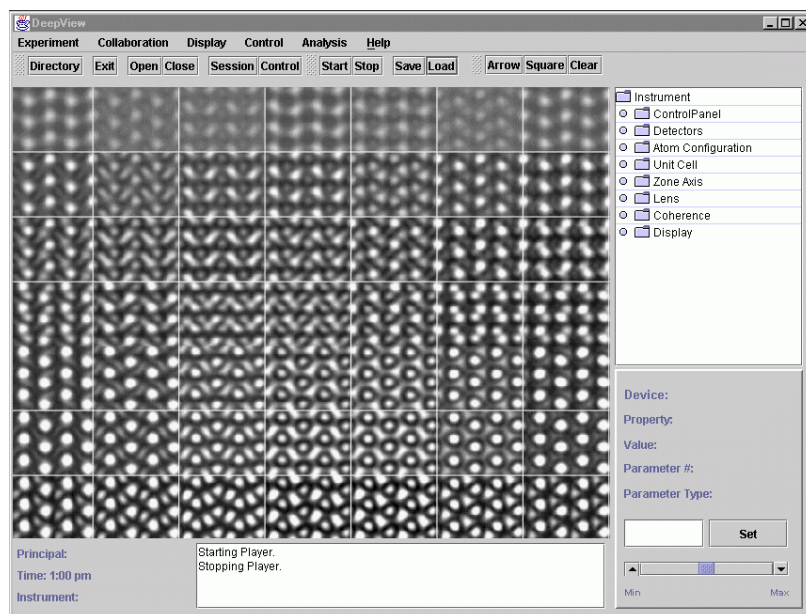


Figure 9: Collage of simulated images at different scales of defocus and thickness.

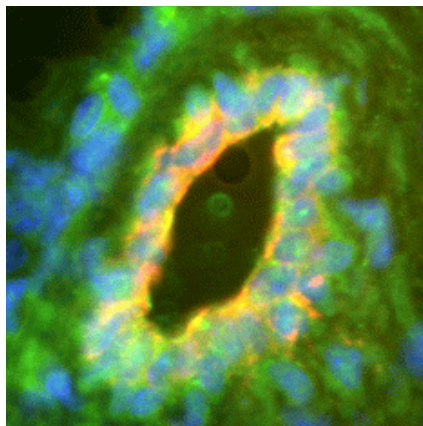


Figure 10: Fluorescence microscopy of a mammary tissue section. Blue corresponds to DNA in cell nuclei, green localizes the latent form of a protein that mediates mammary growth, while orange indicates the presence of the active form of this protein in the epithelium.

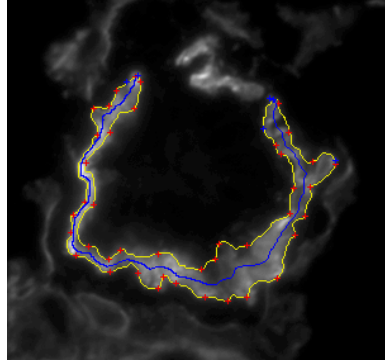


Figure 11: Structural and functional summary of the radiation-induced basement membrane. The yellow line and red dots correspond to structural summary, the blue line encodes the functional representation of the microenvironment (protein expression along the tissue).

recovering corresponding boundaries, and localizing concave corners from their polygonal approximation. These corners provide possible cues to where two adjacent nuclei may touch each other. Thresholding separates big clumps consisting of several nuclei squeezed together, but misses the boundaries between adjacent nuclei. However, the partial boundary between adjacent nuclei can be recovered by extracting crease segments. These crease segments correspond to trough edges (positive curvature maxima), but false creases may also be extracted in the process. Nevertheless, dealing with noise and erroneous segments is a higher-level process that is handled during the grouping process. Our system generates a number of hypotheses for possible grouping of the boundary segments. A unique feature of this system is in hyperquadric representation [8] of each hypothesis and the use of this representation for global consistency. The main advantage of such a parameterized representation (as opposed to polygonal representation) is compactness and better stability in shape description from partial information. In this fashion, each step-edge boundary segment belongs to one and only one nucleus, while each roof-edge boundary segment is shared by two and only two nuclei. These initial hypotheses and their localized inter-relationship provide the basis for the search in the perceptual grouping process. This is expressed in terms of an adequate cost function and minimized through dynamic programming. The final result of this computational step is then shown to an operator for verification and elimination of false alarms. An example of segmentation is shown in Figure 12. This system allows structural information such as size, circularity, etc., to be computed. Furthermore, each nucleus is assigned a unique identifier mask that is used to compute protein expression from a secondary channel.

4 Conclusion

A channel for distributed microscopy has been implemented to meet the requirement for synchronous and asynchronous collaboration. We have leveraged OMG-defined services and proposed three additional services for instrument control, collaborative management, and analytical capability. Our approach aims to leverage common off-the-shelf middleware software to exploit the economy of scale. However, a key design feature of our system has been scalability. This is achieved by advertising instrument and analytical software properties so that the same interface can be reused for different applications. Another feature of our system is the close integration of data collection with

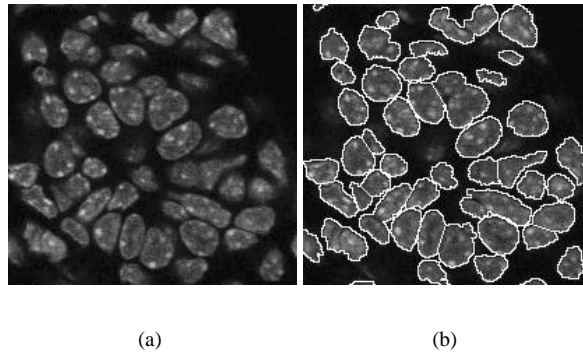


Figure 12: Segmentation results: (a) original image; (b) delineation results.

online data analysis, annotation, and storage. Furthermore, we have shown that a simulation engine behaves much like a scientific instrument. Thus, the same design concepts can be reused to improve the cost-benefit ratio.

Acknowledgments: The authors thank U. Dahmen, W. Bethel, J. Mabone, J. Rome, N. Zaluzec, E. Vogel, E. Kenik, M. Wright, J. Hunt, C. Meyer, and G. Fontenay for valuable discussions.

References

- [1] C. Baru, R. Frost, R. Marciano, R. Moore, A. Rajasekar, and M. Wan. Metadata to support information-based computing environment. In *IEEE Conference on Meta Data Computing*, 1997.
- [2] G. Cong and B. Parvin. Shape from equal thickness contours. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, 1998.
- [3] G. Cong and B. Parvin. Shape from interference patterns. In *Proceedings of the International Conference on Pattern Recognition*, 1998.
- [4] G. Cong and B. Parvin. Model based segmentation of nuclei. In *IEEE Conference on Computer Vision and Pattern recognition*, pages 256–262, 1999.
- [5] The Object Managment Group. <http://www.omg.org>.
- [6] M. Hadida-Hassan, S. Young, and et al. Web-based telemicroscopy. *Journal of Structural Biology*, 125:235–245, 1999.
- [7] N. Kisseberth, G. Brauer, B. Grosser, C. Potter, and B. Carragher. Javascop: A a web-based tem control interface. *Journal of Structural Biology*, 125:229–234, 1999.
- [8] S. Kumar, S. Han, D. Goldgof, and K. Boeyer. On recovering hyperquadrics from range data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(11):1079–1083, 1995.

- [9] S. McCanne. Scalable multimedia communication using ip multicast and lightweight sessions. *IEEE Internet Computing*, 3(2):33–44, 1999.
- [10] M.A. O’Keefe, P.R. Buseck, and S. Iijima. Computed crystal structure images for high resolution electron microscopy. *Nature*, 274:322–324, 1978.
- [11] B. Parvin and et al. Telepresence for in-situ microscopy. In *IEEE Int. Conference on Multimedia Systems and Computers*, Japan, 1996.
- [12] B. Parvin, C. Peng, W. Johnston, and M. Maestre. Tracking of tubular molecules for scientific applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17:800–805, 1995.
- [13] B. Parvin, J. Taylor, D. Callahan, W. Johnston, and U. Dahmen. Visual servoing for on-line facilities. *IEEE Computer*, 1997.
- [14] B. Parvin, J. Taylor, and G. Cong. A service based framework for distributed microscopy. In *IEEE Conference on High Performance Computing and Networking*, 1998.
- [15] B. Parvin, J. Taylor, and G. Cong. Deepview: A system distributed microscopy. In *International Symposium on Distributed Object Applications*, 1999.
- [16] C. Potter and et al. Leginon: a system for fully automated acquisition of 1000 electron micrographs a day. *Ultramicroscopy*, 77:153–161, 1999.
- [17] C. Potter and et al. Evac: A virtual environment for control of remote imaging instrumentation. *IEEE Computer Graphics and Applications*, pages 62–66, 1996.
- [18] The Globus Project. <http://www.globus.org>.
- [19] D. Schmidt, D. Levin, and S. Mungee. The design of the tao real-time object request broker. *Computer Communications*, 21, 1998.
- [20] D. Schmidt and S. Vinoski. Object interconnections—the OMG event services. *SIGS C++ Report*, 1997.
- [21] Young S.J. and et al. Implementing collaboratory for microscopic digital anatomy. *Int. Journal of Supercomputer Applications and High Performance Computing*, pages 170–181, 1996.
- [22] S. Subramanian and et al. Software architecture for the uarc web-based collaboratory. *IEEE Internet Computing*, 3(2):46–51, 1999.